2D arrays Lab

1) Declare and create a 2-dimensional array of floats named numbers and fill it with randomly generated values.

```java
float[][][] numbers = new float[10][20][30][40];

for (int i = 0; i<numbers.length; i++) {
    for (int j=0; j<numbers[i].length; j++) {
        numbers[i][j][k][l] = random(500);
    }
}
```

2) Modify your answer to 1) so that numbers is created as a 4-dimensional array of floats and fill it with randomly generated values.

```java
float[][][][] numbers = new float[10][20][30][40];

for (int i = 0; i<numbers.length; i++) {
    for (int j=0; j<numbers[i].length; j++) {
        for (int k = 0; k <numbers[i][j].length; k++) {
            for (int l = 0; l < numbers[i][j][k].length; l++) {
                numbers[i][j][k][l] = random(500);
            }
        }
    }
}
```

3) Modify your answer to 2) so that the array numbers is created as a ragged 4-dimensional array instead. Only the last dimension needs to be ragged. Use random integers for the lengths of the ragged rows.

```java
float[][][][] numbers = new float[10][20][30][];

for (int i = 0; i<numbers.length; i++) {
    for (int j=0; j<numbers[i].length; j++) {
        for (int k = 0; k <numbers[i][j].length; k++) {
            numbers[i][j][k] = new float[int(random(1,40))];
            for (int l=0; l<numbers[i][j][k].length; l++) {
                numbers[i][j][k][l] = random(500);
            }
        }
    }
}
```
4) Modify your answer to 3) so that the array `numbers` is created as a ragged 4-dimensional array, and all dimensions are ragged. Use random integers for the lengths of all rows.

```java
float[][][][] numbers = new float[int(random(1,10))][]][][];
for (int i = 0; i<numbers.length; i++) {
    numbers[i] = new float[int(random(1,20))][]][];
    for (int j=0; j<numbers[i].length; j++) {
        numbers[i][j] = new float[int(random(1,30))][];  
        for (int k = 0; k <numbers[i][j].length; k++) {
            numbers[i][j][k] = new float[int(random(1,40))];
            for (int l=0; l<numbers[i][j][k].length; l++) {
                numbers[i][j][k][l] = random(500);
            }
        }
    }
}
```
1) Consider the following method. Describe the value returned by a call to this method.

```java
int mystery(int[][] numbers, int val) {
    int idx = -1;
    for (int i = 0; i < numbers.length; i++) {
        for (int j = 0; j < numbers[i].length; j++) {
            if (numbers[i][j] > val) {
                idx = j;
            }
        }
    }
    return idx;
}
```
2) Write a function `int maxSum(int[][] matrix)` which determines which row or column in the 2D array `matrix` has the maximum sum and returns it (the sum). You may assume `matrix` is square.

```java
int maxSum(int[][] matrix) {
    int[] sums = new int[matrix.length*2];
    for (int i=0; i<matrix.length; i++) {
        for (int j=0; j<matrix[i].length; j++) {
            sums[i]+=matrix[i][j];
        }
    }
    for (int i=0; i<matrix.length; i++) {
        for (int j=0; j<matrix[i].length; j++) {
            sums[matrix.length+i]+=matrix[j][i];
        }
    }
    return max(sums);
}
```

3) Write a function `int[][] transpose(int[][] matrix)` which returns the transpose of the input 2D array `matrix`. Recall that the transpose T of a matrix M is defined such that T[i][j] = M[j][i], for all i and j. You may assume `matrix` is square.

```java
//assuming a square matrix
int[][] transpose(int[][] matrix) {
    for(int i=0; i<matrix.length; i++) {
        for(int j=i; j<matrix[i].length; j++) {
            int tmp = matrix[i][j];
            matrix[i][j]=matrix[j][i];
            matrix[j][i] = tmp;
        }
    }
    return matrix;
}
```

4) Write a function `PImage select(int x, int y, int s)` which takes an x and a y screen coordinate and returns an image that is s by s in size and contains the pixels that make up the s by s neighborhood around (x, y). For example, `select(mouseX, mouseY, 10)` will return a 10 by 10 pixel region that surrounds the current mouse location. (In the case where s is even, there should be more pixels to the left and above the mouse position.)

```java
PImage select(int x, int y, int s) {
    return get(x-s/2, y-s/2, s, s);
}
```